Simulació d'un algorisme de navegació sobre un model i ús dels resultats en un robot real

Alex Ramajo Teixidó

Resum— Aquest projecte està pensat per implementar un robot de forma virtual utilitzant un framework creat per al desenvolupament de robots. El desenvolupament ha passat a través de les fases de disseny de prototips únicament 3D usant *blender*, implementació del Robot usant *Unified Robot Description Format amb l'ajuda de RViz*, importació del model en el entorn de simulació Gazebo, programació del moviment amb C++, redirecció de la interfície de moviment de *Turtlebot* i integració de sensors. El objectiu final del desenvolupament del projecte és obtenir un robot manipulable en un entorn simulat i obtenir dades d'aquest entorn mitjançant sensors com càmeres o làsers.

Paraules clau — Robotica, 3D, Simulació, Programació, Disseny, Moviment

Abstract— This project is designed to implement a robot virtually using a framework made for the development of robots. The development has gone through phases of design 3D prototypes only using Blender, implementation Robot using Unified Robot Description Format with the help of RViz, import the model into the simulation environment Gazebo, programming motion with C ++, redirection of Turtlebot's movement interface and integration of sensors. The ultimate goal of the development project is to obtain a manipulatable robot in a simulated environment and obtain data in this environment through sensors and cameras or lasers.

Index Terms — Robotics, 3D Simulation, Programming, Design, Movement

– – – – – – – – – – ◆ – – – – – – – – – –

1 Introducció

El món de la robòtica avui en dia està molt present en les nostres vides i cada cop ho estarà més donades les avantatges que dona el ús d'aquesta tecnologia. Tasques molt repetitives, perilloses o que simplement un humà no pot fer poden ser realitzades per robots. Aquests robots primer s'han de dissenyar tenint en compte molts factors tant físics com algorítmics. El procés de disseny d'aquestes màquines ha evolucionat molt gracies a que ara comptem amb simuladors que permeten provar moltes possibilitats abans de donar el disseny per acabat.
En part aquesta evolució és gracies al framework ROS, software específicament dissenyat per al estudi i desenvolupament d'elements relacionats amb la robòtica.

1.1 Objectius

Els objectius a assolir en el projecte són els següents:

- Familiaritzar-se amb el software ROS i els seus components principals:
  - Unified Robot Description Format
  - Gazebo
  - RViz
- Dissenyar un robot complert
  - Dissenyar el Robot en 3D
  - Dotar de moviment al robot
  - Implementar sensors

1.2 Metodologia utilitzada

Durant les primeres setmanes del projecte es va pretendre entendre ROS i el seu funcionament intern, i per a aquest objectiu va ser de gran ajuda el llibre *Mastering ROS for Robotics Programming* de *Lentin Joseph*. El llibre s'ha fet server com a guia durant tot el projecte ja que aporta molta informació.
La següent fase va ser realitzar diferents models 3D amb el software Blender "en brut" per veure quin s'adaptava més a les nostres necessitats. Un cop vam escollir un prototip, es va passar a implementar amb URDF, amb l'ajuda de RViz i Gazebo.

- *E-mail de contacte: Alex.Ramajo@outlook.com*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Ricardo Toledo (Computació)*
- *Curs 2016/17*

Un cop el model estava ja implementat, es va passar a donar-li mobilitat, i donant la possibilitat de controlar-lo des de teclat.
I ja finalment, se li va dotar de una càmera.

2 ESTAT DE L'ART

La indústria de la robòtica està cada dia més present tant en les nostres vides com en el mon laboral ja que dona noves possibilitats. L'evolució dels ordinadors que cada cop son més ràpids ajuda directament a que evolucioni la robòtica. Sobretot la velocitat de processament de dades ja que fa els robots més intel·ligents i per tant poden realitzar més tasques.

La seva utilització és massiva avui en dia i es fa servir per tasques tant diverses com pot ser una cadena de muntatge, o exploracions d'altres planetes. Per això, és impossible plantejar-se un futur sense robots.

3 PROPOSTA INICIAL I COM HA EVOLUCIONAT

Al llarg del desenvolupament del projecte, aquest s'ha vist modificat i en aquest apartat explicarem com.

2.1 La idea inicial

El projecte inicial era el de desenvolupar un algoritme de navegació autònoma, programant-lo sobre un robot virtual en un entorn simulat com Gazebo. Al treballar en un entorn virtual, teníem total llibertat per provar moltes més alternatives que si ho féssim directament sobre el robot, ja que és molt més ràpid i no cal pujar codi nou a la màquina física cada vegada que es vulgui provar. També ens permet una menor dependència del hardware i per tant no necessitem un entorn concret per treballar, ho podem fer en qualsevol lloc.

Un cop el algoritme funcionés de manera correcte en el simulador, es passaria a provar en un robot real, en concret un turtlebot.

2.2 La idea final

Vam voler ampliar el abast del treball afegint també el disseny 3D complet del robot. Això implicava pensar la forma que volíem, modelar-lo, implementar les físiques i el moviment, i finalment afegir el sensor o sensors que necessitéssim.

Finalment, la tasca de desenvolupar el robot sencer des de 0 ha portat més hores de les que pensàvem inicialment i no hem arribat a aplicar el algorisme de navegació, tot i que de disposar de alguna setmana més de temps hauria estat possible.

4 ROBOT OPERATING SYSTEM **"ROS"**

ROS és un paquet software pensat i creat per a tot allò relacionat amb el món de la robòtica. Permet dissenyar un robot des de 0 abans de implementar-lo físicament. Això atorga avantatges que no serien possibles amb altres metodologies de desenvolupament.

3.1 Cost 0 en anar endarrere:

Una de les avantatges més importants és el fet de que sempre pots realitzar canvis en fases anteriors del desenvolupament sense patir pèrdues econòmiques. Per exemple:

A. Metodologia tradicional: El model del robot ja està fet, es fabriquen les peces i es realitza el muntatge d'aquest. Un cop muntat, es veu que el robot és poc estable donada la seva morfologia.

En aquest cas, s'hauria perdut temps en la fabricació de les peces, a més de diners, ja que aquestes peces no serien valides. Caldria tornar a la fase de disseny, mirar de modificar-lo de manera que el robot fos més estable, tornar a fabricar les parts, muntar-lo i provar si s'ha solucionat el problema.

B. Metodologia amb ROS: Ens trobem en la mateixa situació que el cas anterior, el model està fet, però en comptes de fabricar les parts, s'exporta el model 3D a un simulador com ara Gazebo.
Gazebo simula un entorn real, i com a simulador implementa per exemple la força de la gravetat que és una de les forces que afecten a la estabilitat del nostre robot.

Amb el model exportat a Gazebo, veuríem en el simulador que el robot és poc estable però sense pèrdues econòmiques ja que no hem necessitat fabricar les parts.
Només ens caldria revisar el model i tornar a provar el robot en el entorn Gazebo, procés que és molt ràpid.

El procés de modificar el model del robot i tornar a provar-lo en gazebo és ràpid, el procés que requereix de més temps és el de adaptar per primera vegada el model per a exportar-lo a Gazebo, com veurem més endavant.

3.2 Funcionament ROS

ROS està basat en una arquitectura de grafs. En els nodes, hi ha els processos mentre que les connexions serien nodes subscrits a altres nodes que en recullen la informació.

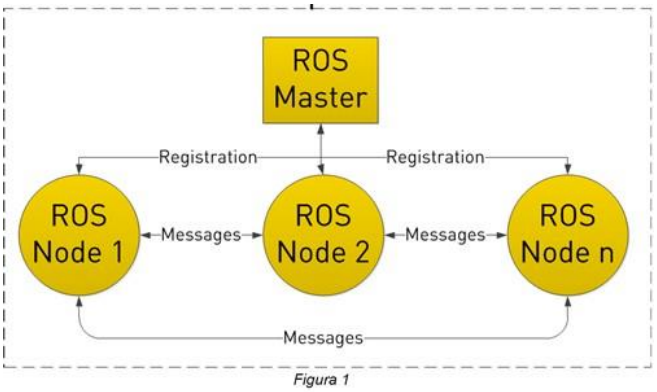


Figura 1

Com podem veure a la figura 1, els nodes estan tots registrats en el ROS Master, i intercanvien missatges entre ells.

Si mirem amb més detall, el que realment està passant és el següent:

Els nodes son processos com per exemple un procés que mitjançant un sensor làser, calcula una distància. Aquesta dada, el node la publicarà en un tòpic per fer-la accessible a altres nodes / processos.

Un altre node, és el encarregat de decidir cap a on s'ha de moure el robot, per tant, necessita aquesta dada que ha publicat un altre node. El que hem de fer llavors, és fer que aquest node es subscrigui al tòpic on està publicant aquesta dada l'altre node. D'aquesta manera, en tot moment aquest node podrà veure aquesta informació.

3.3 Unified Robot Description Format - URDF

Per a introduir el nostre model de robot a ROS hi ha diverses maneres. ROS és compatible amb els més populars formats d'arxius 3D, i per tant podríem fer el model fent ús d'un software de disseny 3D com per exemple Blender o Cinema 3D.
Per a fer ús d'aquests softwares, cal un mínim de coneixement o experiència en el mon del disseny 3D, i per intentar no crear aquest impediment a l'hora de dissenyar un robot, ROS compta amb el llenguatge URDF.

El Unified Robot Description Format o URDF és un llenguatge basat en tags que permet descriure físicament un robot.

Hi ha 2 tags principals amb els quals es lliga tot el model, aquests tags son *Link* i *Joint*:

3.3.1 Link

Amb el tag link es descriuen les parts físiques del robot, com un braç o la base.

El tag link ens permet descriure cossos rígids i les seves característiques visuals com el color, o característiques de comportament físic com el camp de col·lisió o la inèrcia de la part.

Per a descriure correctament una peça amb el tag link, ens caldrà especificar dins d'aquest un nom amb el camp name, que és obligatori dins el tag link.

A continuació es pot veure una llista amb els diferents elements que pot contenir el tag link:

A. Atributs: El tag link només té com atribut obligatori el camp Name

TAULA 1
Atributs del tag *link*

Atribut	Descripció
Name	El nom del link

B. Elements: Els elements son tots opcionals*

TAULA 2
Elements del tag *link*

Element	Descripció
Inertial	El nom del link
Visual	Propietats visuals del link. Permet especificar el aspecte del link
Collision	Crea el camp de col·lisió del link

A continuació explicarem amb més detall cada un d'aquests elements.

1) Element Inertial (opcional)

El tag inertial ens permet, tal i com el seu nom indica permet descriure el comportament inercial de la part o peça que estiguem definint. Per a fer-ho, tenim 3 camps:

Origin (opcional) :

Indica on està el frame de referència de la inèrcia respecte al frame de referència del link. Per posicionar correctament aquest frame, ho farem amb els camps

xyz: posicionament x, y i z del frame d'inèrcia respecte al frame del link.

rpy: orientació del frame d'inèrcia fent servir els eixos roll, pitch y yaw, sempre en radians.

En la figura 2 podem veure un frame amb els seus respectius *x,y,z* i *r,p,y*.

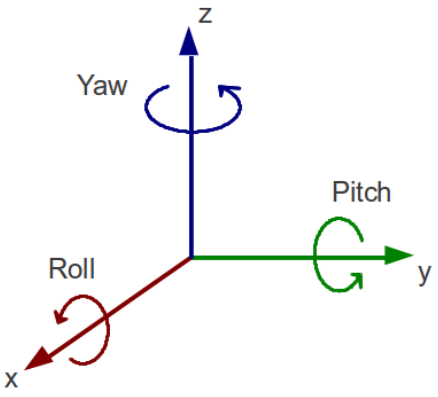


Fig. 2. Frame amb les seves variables de posicionament i rotació així com també els colors que es solen associar a aquests eixos.

Quan parlem de frames respecte a altres frames, el que significa és que la posició d'un frame, és a dir l'increment o decrement de les seves variables x, y i z es realitzen a partir del frame anterior actuant com a 0 global del sistema. Això ajuda a simplificar el posicionament de nous frames.

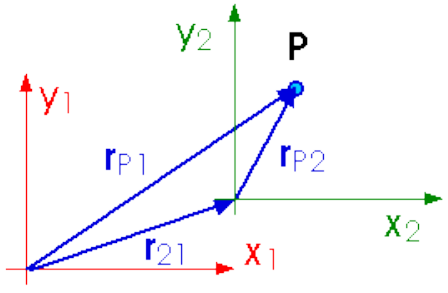


Fig. 3. Exemple de un frame posicionat respecte a un altre frame on es poden veure els eixos de translació.

Mass:

Defineix la massa del link, en kilograms.

Inertia:

Per definir la inèrcia del link es fa amb una matriu rotacional 3x3.

$$I_p = \begin{vmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & I_{-yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{vmatrix}$$

Fig. 4. Només cal definir els 6 valors superiors de la matriu.

Al tractar-se d'una matriu simètrica diagonalment, només cal definir els 6 valors superiors d'aquesta, com podem veure a la Fig. 4.

2) Element Visual (opcional)

El tag visual especifica com serà la forma del link o objecte, i el conjunt de links amb camp visual especificat formaran la forma final del robot. Per definir correctament el aspecte del link tenim els següents camps:

Name (opcional) : Serveix per donar nom a una part física del link. Ja que és possible definir múltiples tags visual dins d'un mateix link, ens permetrà diferenciar entre ells.

Origin (opcional) :

Indica el frame de referencia del element visual respecte al frame de referencia del link, és a dir, ens permet posicionar visualment un element com per exemple un braç, respecte a la posició del link. Com sempre que volem especificar un punt d'origen, haurem de definir els paràmetres

- xyz: posicionament x, y i z del frame visual respecte al frame del link.
- rpy: orientació del frame fent servir els eixos roll, pitch y yaw.

Geometry :

Ens permet definir la forma visual que tindrà el link que hem creat i es pot inicialitzar d'una de les següents maneres.

- Box: la forma inicial serà un cub, i li podem especificar el atribut *size* dels 3 eixos *x, y i z*. Modificant aquets atributs, podrem formar altres figures com rectangles per exemple.
- Cylinder: podrem crear un cilindre, especificant els seus atributs *radius* i *length.*
- Sphere: crearem una esfera i podem modificar la seva mida amb el atribut *radius.*

- Mesh: ens permet importar una forma predefinida en un altre software en diferents formats de 3D com per exemple *.stl*. Podem modificar només la seva variable *size*.

Material (opcional) :

Podem aplicar un material a un link, és un element visual que pot donar textura i així dotar de més realisme al model 3D del nostre robot.

Per especificar un material ho podem fer amb

- Name: Nom del material
- Color (opcional)
- Texture (opcional) : Es pot aplicar una textura al material a partir d'un arxiu.

Fig. 5. Exemple de textura metàl·lica

3) Element Collision (opcional)

El element Collision d'un link és el que fa que aquest interactuï de manera física amb el entorn. Per exemple, que un link no travessi parets o altres links.
Les col·lisions les podem tractar de manera simple o complexa depenent del grau de precisió que necessitem.

Per exemple, en la Fig. 6. podem observar un model en 3D d'un sofà en el motor Unreal Engine amb la seva malla de col·lisió simple en color morat i la malla de col·lisió complexa en color blau.

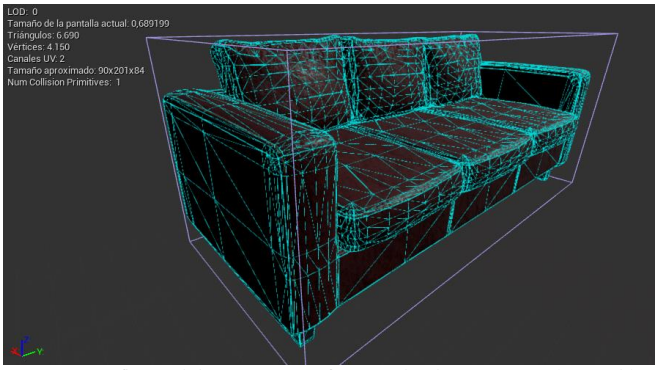


Fig. 6. Sofà modelat en un software de disseny 3D (zBrush) i posteriorment exportat a Unreal Engine. El model en concret està format per uns 7000 triangles. Fer els càlculs per a les col·lisions de tants triangles és molt costós per la CPU, i per tant es crea una malla de col·lisió simple (malla exterior en color morat), de només 6 costats molt fàcil de calcular

Per a definir la malla de col·lisió en URDF tenim els següents camps:

- Name (opcional) : Ens permet diferenciar entre diferents malles de col·lisió dins del mateix *link*.
- Origin (opcional) : Permet indicar on volem en frame de col·lisió relatiu al frame del *link*
  - xyz: posicionament x, y i z del frame respecte al frame del link.
  - rpy: orientació del frame fent servir els eixos roll, pitch y yaw.
- Geometry : Exactament igual que en el tag visual, ja que la malla de col·lisió no deixa de ser una malla igual que la geometria del link en sí mateixa. La diferència és que una té propòsits visuals i l'altre de propietats físiques.

En ROS és important no abusar de malles de col·lisió molt elaborades i fer-les el més simple possible per tal de no sobrecarregar la CPU amb masses càlculs que a la hora de la veritat, no son necessaris.

Amb els elements que hem vist fins ara podem definir un *link* o part de robot. Si en definim múltiples, podrem formar un robot sencer. Una forma de veure gràficament els 3 elements que s'han explicat ara és la que ens mostra la Fig. 7.

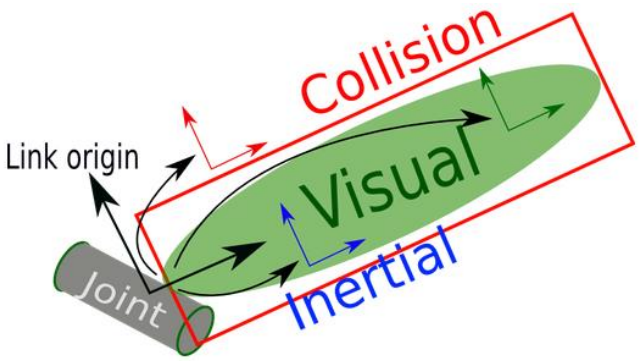


Fig. 7. Veiem un diagrama que mostra els 3 elements d'un link: Inertial, Visual i Collision. El link està connectat a un joint, del qual ara en parlarem.

Els *links* permeten descriure físicament un robot de manera completa, però per tindre un robot completament funcional és necessari afegir *Joints* a aquest robot que lliguin els *links* i així dotar de mobilitat al conjunt.

3.3.2 Joint

Els joints son elements que serveixen per descriure les cinemàtiques o moviments que poden realitzar 2 *links* entre ells.

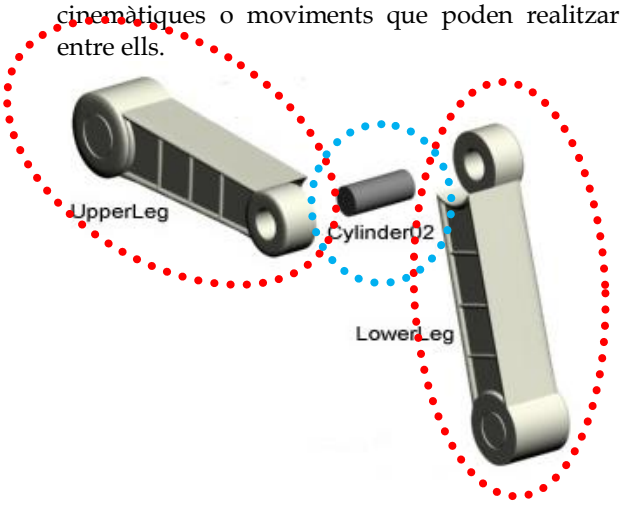


Fig. 8. Exemple de joint entre dos links. Els links estan encerclats de color vermell i el joint en blau.

Si mirem la Fig. 8. veurem que tenint dos links encerclats de color vermell, i que els estem posant entre ells un joint, que està encerclat de color blau.

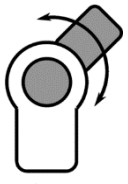
Un cop aquesta unió sigui efectiva, ens quedarien els 2 links units i es podrien moure un respecte de l'altre.
En aquest cas la unió quedaria com podem veure a la Fig. 9. i això es tractaria d'un joint de revolució.

Fig. 9. Resultat de la unió de 2 links amb un joint. És el resultat del exemple vist a la Fig. 8.

Amb aquesta configuració, tindrem un únic moviment, però si haguéssim fet servir un altre tipus de joint per a realitzar la unió, en podríem obtenir altres.

Tenim diversos tipus de *joints*, parlarem dels més comuns a continuació:

- Revolute: Permet realitzar un gir sobre un eix i que té com a límits els propis links físics que no pot travessar.



Revolute joint

- Continuous: És el mateix cas que l'anterior però no tenim aquest límit normalment perquè les peces no coincideixen.
- Prismàtic: Es tracta d'una part que llisca sobre una altre en una determinada direcció o eix. Es tracta d'un moviment de translació.
- Fixed: No permet moviment, però sí lligar dos links entre si.

Passem a veure els atributs i elements que tenim amb el tag *joint*.

A. Atributs: Tenim dos atributs obligatoris

TAULA 3
Atributs del tag *joint*

Atribut	Descripció
Name	Nom del joint
Type	Especificarem el tipus de joint

B. Elements:

Tenim molts elements per descriure el *joint* ja que es tracta de parts del robot més complexes que requereixen de més informació, si els comparem amb els links. A continuació veurem els més importants.

TAULA 4
Elements del tag *joint*

Element	Descripció
Origin	Posició del joint respecte el frame del pare. Marca la posició on comença el link child. Fig. 10.
Parent	Link pare del *joint*
Child	Link fill del *joint*
Axis	Permet definir sobre quin eix actuarà el *joint*.
Calibration	Posicions de referència del *joint* que serveixen per calcular la seva posició absoluta
Dynamics	Descriu propietats físiques del *joint*
Limit	Límits físics del *joint*

Com hem fet amb el tag *link*, ara s'explicaran amb més profunditat aquests elements i les opcions que ens donen.

1) Element Origin (opcional)

Permet posicionar el joint respecte al parent, i per tant també el child. D'aquesta manera posicionar el child és senzill ja que es col·locarà seguidament del joint com es pot veure a la Fig. 10.
Per posicionar-lo, tenim els elements:

- xyz: posicionament x, y i z del joint visual respecte al frame del parent.
- rpy: orientació del joint fent servir els eixos roll, pitch y yaw.

2) Parent

Per definir un *joint,* cal especificar el *link* pare d'aquest joint. Ho farem simplement escrivint el nom del link.

- Link: Nom del link pare

3) Child

També cal definir el link fill o child.

- Link: Nom del link fill

4) Axis (opcional) :

Permet definir el eix sobre el que actuarà el *joint*, per exemple si és un *revolute* joint, sobre quin eix girarà aquest *joint*.

- xyz: Representen els eixos, i posarem a 1 el eix que volem que sigui sobre el que s'actua.
  P.e. (0 0 1)

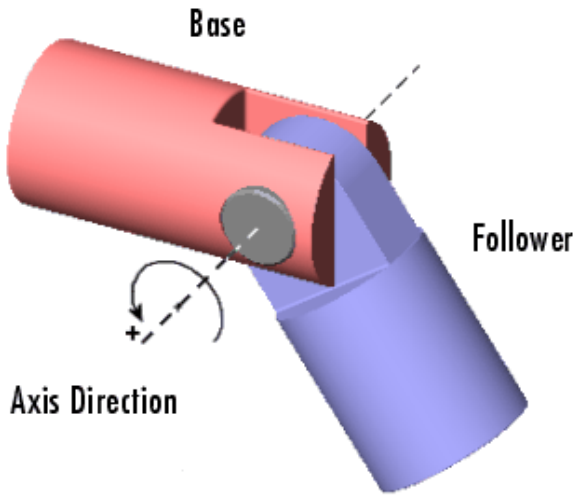


Fig. 10. Es pot observar un revolute joint amb el seu eix de rotació

3.4 Gazebo y RViz

Gazebo i RViz son complements indispensables en el sistema ROS, donat que sense ells seria molt complicat desenvolupar el robot.

RViz ens proporciona un entorn de visualització centrat en el robot, i ens permet anar veient els progressos que realitzem en el model 3D del nostre robot mentre l'anem implementant amb URDF. En la Fig. 11 podem veure la seva interfície.
Només necessita que l'arxiu URDF contingui els links i joints que descriuen el robot i ja el podrà renderitzar correctament.

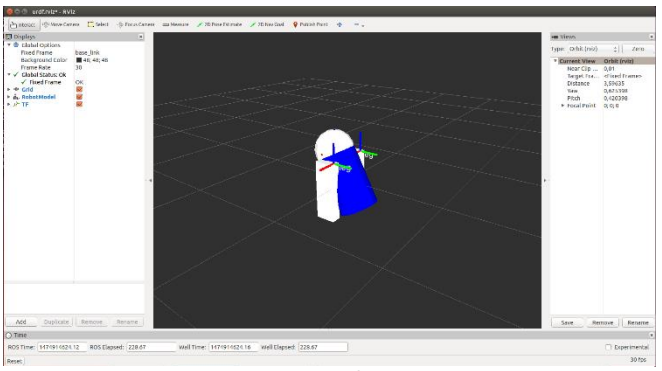


Fig. 11. Interface RViz

Gazebo per altra banda es tracta d'un simulador on un cop completat el nostre disseny del robot amb URDF i afegint un seguit de nous tags ens permetrà veure com es comporta el robot sota les lleis físiques del nostre planeta, com per exemple la gravetat. Això permet comprovar l'estabilitat del robot i corregir-la en cas de ser necessari. Gazebo també permet importar entorns o crear-los en el propi simulador, i simular-hi per exemple un algoritme de navegació autònoma. A la Fig. 12. es pot veure la interfície de Gazebo.

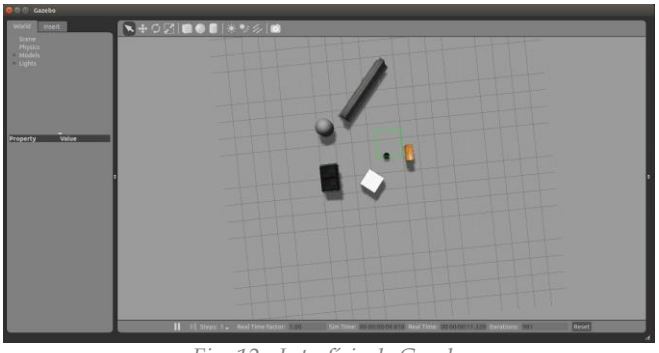


Fig. 12. Interfície de Gazebo

4 EL ROBOT

Volíem un robot amb bona mobilitat però no molt complex, i a poder ser amb una fisiologia que facilités actualitzar-lo amb nous sensors en el cas de que es volgués. El primer prototip va ser una base rectangular amb 4 rodes que es pot veure a la Fig. 13. :

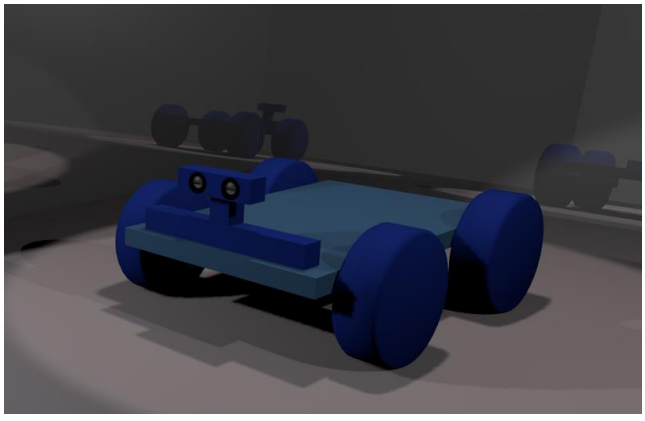
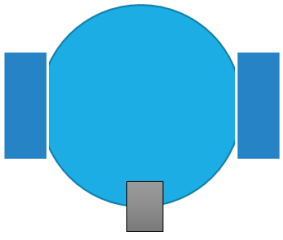


Fig. 13. Primer model prototip realitzat en blender. Té quatre rodes motoritzades i una base. Sobre la base hi ha una estructura que permet incloure sensors.

Aquest model tot i ser funcional, no donava prou joc o complexitat a la hora de ser implementat en URDF, a més de que caldrien 4 motors, 1 per roda i això el faria car. Un mateix resultat però amb 2 motors és possible si agafem com a exemple un turtlebot. Es tracta d'un robot format per una base, dos rodes motoritzades i 1 roda més de suport a la part frontal.



Si el volem ampliar, també seria força senzill ja que només caldria afegir tres columnes de suport, i es podria muntar una segona base a sobre.

Seguint la configuració anterior, el prototip escollit finalment per implementar en URDF va estar el que es pot veure en la Fig. 14.

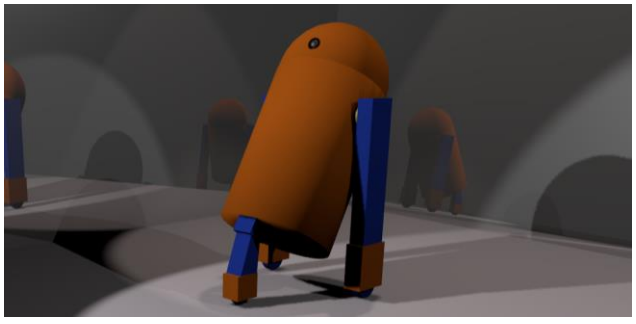


Fig. 14. Model 3D renderitzat en Bleder, prototip que s'ha implementat en URDF.

El cilindre central actua com a base que connecta les rodes motores amb la roda de suport, i a la part superior té espai per sensors.

Aquesta configuració ens dona la mobilitat que volem, ja que pot realitzar tota mena de girs i podem afegir sensors.

El següent pas va ser implementar el prototip en URDF. A continuació en la Fig. 15. es pot veure el resultat obtingut junt amb la topologia.

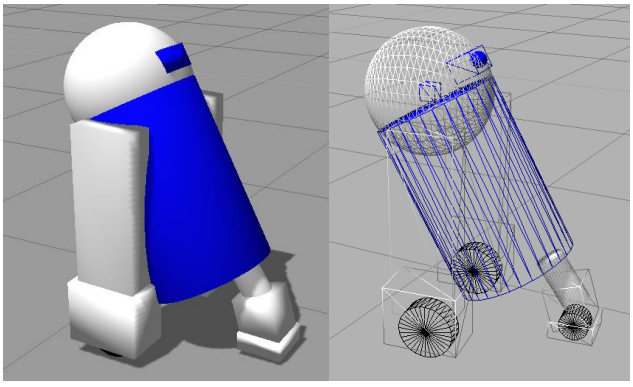


Fig 15. A l'esquerra podem veure el model en urdf renderitzat ja en gazebo, i a la dreta la topologia del robot per veure millor les rodes i l'estructura.

Si volem veure la configuració dels joints de les rodes, Gazebo permet activar els eixos d'aquests per visualitzar-los com es pot veure a la Fig. 16.

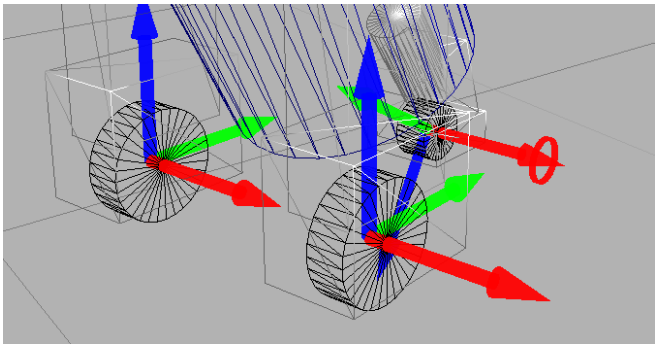


Fig. 16. Joints de les rodes de tipus contiuous, que per tant poden girar sense límit en les 2 direccions sobre l'eix X (vermell). La roda davantera recordem que no es motoritzada

Amb el robot correctament modelat, s'ha de connectar amb ROS. Per fer-ho comptem amb plugins de gazebo que actuen com a interfície entre el robot modelat i ROS. Per exemple, per controlar les rodes, el primer que cal fer és carregar el plugin "*gazebo_ros_control*" i definir controladors per cada una de les rodes motoritzades.
Les dades d'aquests joints, com la velocitat o posició es publicaran (Fig. 17.) per tal que altres nodes ROS puguin veure-les i actuar-hi a sobre.

Amb els controladors de les rodes publicats, ja és possible fer moure el robot manualment enviant missatges a aquests tòpics, però el control no és còmode, ja que només pots controlar els joins de manera individual. Per resoldre això vam afegir un plugin anomenat "*differential drive*" que permet que puguem controlar les 2 rodes alhora.

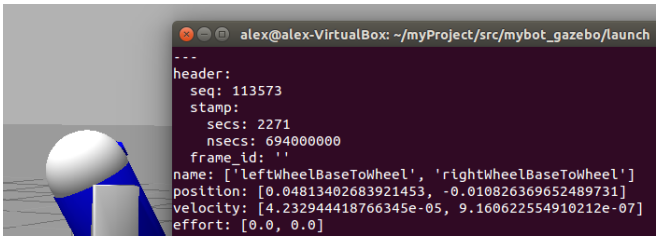


Fig. 17. Podem consultar el estat dels joints en un terminal. Les dades es publiquen en un tòpic de ROS.

Aquest plugin a més ens aporta dades de odometria. Les dades de odometria son indispensables si planegem que el nostre robot es mogui de manera autònoma, ja que ajuden a posicionar el robot respecte al punt de partida.
En el *differential drive* s'especifica els 2 joints que formen part del mateix, i el tòpic al que es subscriu per a poder rebre dades, en aquest cas el tòpic es cmd_vel.

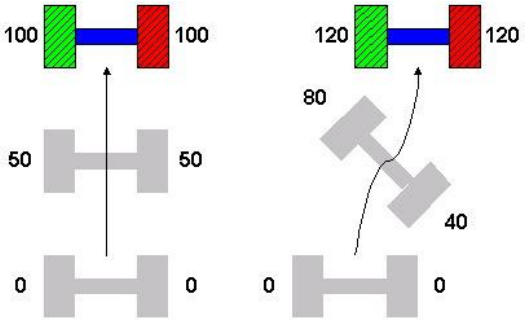


Fig. 18. Les dades de odometria calculen quant de moviment hi ha hagut, basant-se en quant han girat les rodes. D'aquesta manera el robot sap posicionar-se en el mapa de la zona i saber on està per continuar el camí cap al punt objectiu.

Per teleoperar el nostre robot, hem fet server una interfície ja existent, en aquest cas la del turtlebot. El que hem de fer és redirigir cap a on enviarà els missatges aquesta interfície, en aquest cas cap al tòpic cmd_vel on està subscrit el *differential drive.*

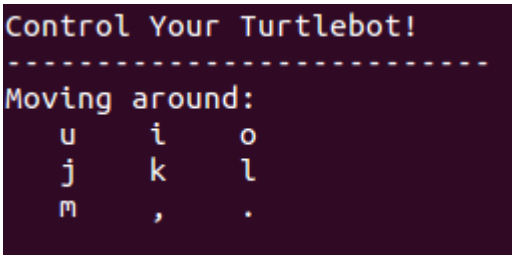


Fig. 19. Interfície per controlar el nostre robot, que a estat adaptada de la seva implementació per al tutrlebot.

En aquest punt ja tenim un robot que podem moure per l'entorn de gazebo, que col·lisiona amb els objectes i proporciona dades de odometria. El següent pas per completar el disseny del nostre robot, és afegir algun sensor que ens proporcioni més dades, en aquest cas vam decidir posar una càmera ja que és un tipus de sensor que obre moltes possibilitats, com per exemple poder teleoperar el nostre robot i veure el que té davant.

Per afegir la càmera primer cal un nou plugin al model del nostre robot. Aquest plugin funciona com un nou link, i per tant primer cal crear un nou *link* anomenat *càmera*. Amb el *link* afegit, podem configurar el plugin *càmera* de gazebo. Com tots els nodes de ROS, aquest plugin publicarà les dades en un tòpic, en aquest cas el tòpic *imatge_raw*. Podem veure el resultat si utilitzem un terminal per subscriure'ns al tòpic fent servir una eina anomenada *imatge_view* que està integrada en ROS.

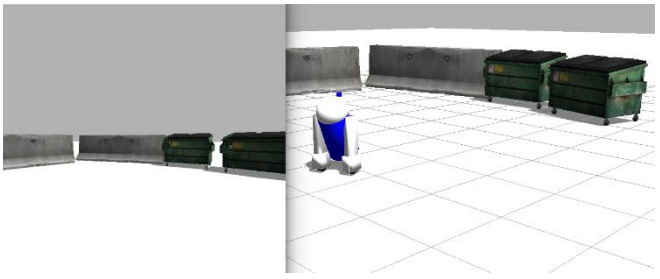


Fig. 20. Amb la càmera configurada podem el robot transmet imatge i podem veure en directe per on anem.

4 Resultats

En aquesta secció es descriurà el resultat final del desenvolupament del projecte junt amb les dificultats que s'han trobat durant la implementació de les diferents etapes de la realització d'aquest.

4.1 Estat actual

En aquest moment el robot és plenament funcional i permet moure'l per un entorn de simulació mentre veiem les imatges que ens transmet per la seva càmera. Podríem dir que la fase de desenvolupament estaria finalitzada, tenint un robot funcional i estable. Ara es podria passar a la fase de fabricació d'un prototip, per exemple aprofitant tecnologies emergents com les impressores 3D per fabricar les seves parts.

4.2 Dificultats

La principal dificultat i que ha estat present durant tot el desenvolupament del projecte ha estat la complexitat del framework ROS, donat que té tantes possibilitats i llibertats. El fet de oferir llibertat total per implementar qualsevol tipus de robot que puguem imaginar, també el converteix en una eina complexa i que és complicat començar.

Una altre dificultat que ens hem trobat ha estat poder importar el nostre disseny del robot en format URDF a Gazebo, donat que s'ha hagut de reestructurar el directori

del projecte i afegir nova informació en el model. com per exemple els tags de inèrcia a cada link.
Per a la correcta simulació de les físiques en gazebo, s'han de posar valors realistes i realitzar moltes probes per comprovar la estabilitat del robot. En el cas del nostre, donada la seva fisiologia tenia el punt de equilibri molt endarrere en la seva estructura, i això causava que quan els motors feien força per fer avançar el robot cap endavant, aquest queia cap enrere. Per solucionar-ho, es va haver de modificar elements com la massa de les parts del robot o les dades de inèrcia a més de diferents paràmetres dels motors.

5 Conclusió

Durant la realització del projecte s'ha treballat amb eines que desconeixíem i com a resultat s'han aprés una sèrie de nous coneixements que ara s'exposaran, a més de diferents línies de treball futures que podrien fer anar el projecte més enllà amb noves característiques.

5.1 Aprenentatge derivat del projecte

En primer lloc he aprés sobre el funcionament del framework ROS i a fer-ne ús per a desenvolupar un robot capaç de moure's i donar-nos informació sobre un entorn a través de diferents sensors. He pogut aprofundir més en el mon de la robòtica i posar en pràctica diversos coneixements adquirits al llarg de la carrera.

Indirectament el projecte ha afectat positivament a la meva capacitat d'aprenentatge autònom i planificació donades les circumstàncies de tractar-se d'una nova tecnologia per mi i d'haver de compaginar la realització del projecte en paral·lel a la meva feina. Tot i disposar de poques hores per dedicar al projecte, gracies a una fase de planificació prèvia hem pogut realitzar-lo amb èxit.

També he pogut aprendre sobre modelació en 3D amb el software blender.

Agraïments

Vull agrair al tutor del projecte Ricardo Toledo per guiarme quan no sabia per on començar, i les alternatives o sortides als problemes o inconvenients que hem anat trobant.

També agrair el suport que he rebut per part de amics i familiars pels ànims i per entendre'm quan no podia dedica'ls-hi el temps que es mereixen.

Bibliografia

[1] ROS, http://www.ros.org/

[2] Gazebo, http://gazebosim.org/

[3] RViz, http://wiki.ros.org/rviz

[4] URDF, http://wiki.ros.org/urdf

[5] Blender, https://www.blender.org/

[6] Turtlebot, http://learn.turtlebot.com/

[7] Robot Sensors, http://www.robotplatform.com/knowledge/sensors/types_of_robot_sensors.html

[8] Mastering ROS for Robotics Programming, *Lentin Joseph, 2015,* Packt Publishing, Ltd.

APÈNDIX A

GLOSSARI

Modelació: Donar forma física a un element, per exemple en 3D.

Odometria: Dades que posicionen a un robot en una posició concreta respecte a un altre. Normalment s'agafa la distància recorreguda per cada roda de manera individual.

Plugin: Part de software que s'afegeix a un cert software ja existent i en millora o augmenta les funcionalitats.

Teleoperar: Controlar un robot a distància o sense entaular contacte directament amb ell.

Triangles (3D) : Unitat bàsica en una estructura 3D. Un conjunt de triangles forma una figura 3D.

APÈNDIX B

EINA DE PLANIFICACIÓ

Per a saber en tot moment en quin punt em trobava i per on continuar es va buscar una eina per ajudar en la planificació de les tasques i documentació de problemes. L'eina que més havia fet servir era Jira, però vaig buscar una alternativa i vaig trobar Trello.

Amb trello vaig posar totes les tasques previstes des d'un inici, i segons en sortien de noves també les afegia, així podia veure més gràficament com anava el projecte.